

Лабораторная работа №2

Задача 1.

Отдача контента через Node.js

Создадим следующую структуру проекта, включающий каталог сайта:

```
site
  img -
    - logo.gif
    - photo.gif
  index.html
  about.html
  contact.html
  style.css
  script.js
server.js
```

Файл index.html содержит следующую разметку

```
<html>
  <head>
    <title>Node-page</title>
    <meta charset="utf-8">
    <link rel="stylesheet" href="style.css" type="text/css">
    <script src="script.js" type="text/javascript"></script>
  </head>
  <body>
    <h1>Сайт на Node.js</h1>
    <div class="main menu">
      <p><a href="about.html">О нас</a></p>
      <p><a href="contact.html">Контакты</a></p>
    </div>
    
  </body>
</html>
```

Согласно наполнению файла index.html создайте остальные файлы в папке site и заполните тестовыми данными по вашему усмотрению (gif картинки скачайте из интернета).

Теперь займёмся наполнением файла server.js.

Сначала нам потребуется подключить модули:

```
let http = require('http'); // подключение модуля для работы с http
let fs = require('fs'); // подключение модуля для работы с файлом
```

Попробуем отдать контент уже опробованным нами способом:

```
http.createServer((request, response) => {
  let pathname = 'site/index.html';
  console.log("Request: " + request.url);
  fs.readFile(pathname, 'utf8', (err, data) => {
    if (err) {
      console.log('Could not find or open file for reading\n');
```

```

        response.statusCode = 404;
        response.end();
    } else {
        console.log(`The file ${pathname} is read and sent to the client\n`);
        response.writeHead(200, {'Content-Type': 'text/html'});
        response.end(data);
    }
});
}).listen(8080, ()=>{
    console.log("HTTP server works in 8080 port!\n");
});

```

Запустите браузер с адресом <http://localhost:8080> и наблюдайте что будет выводиться в консоли? Почему так происходит?

Задача 2.

В предыдущем Задаче 1 представлено все разнообразие контента, которое нам надо в нужном формате отдать браузеру. Если бы этот контент состоял только из html страниц, можно было бы ограничиться следующим кодом:

```

http.createServer((request, response) => {
    let pathname;
    console.log("Request: " + request.url);
    if (request.url === '/')
        pathname = 'site/index.html';
    else
        pathname = 'site' + request.url;
    fs.readFile(pathname, 'utf8', (err, data) => {
        if (err) {
            console.log('Could not find or open file for reading\n');
            response.statusCode = 404;
            response.end();
        } else {
            console.log(`The file ${pathname} is read and sent to the client\n`);
            response.writeHead(200, {'Content-Type': 'text/html'});
            response.end(data);
        }
    });
}).listen(8080, ()=>{
    console.log("HTTP server works in 8080 port!\n");
});

```

Здесь у нас присутствуют не только javascript- и CSS-файлы, которые современный браузер, поморщившись, примет и как 'text/html', но и изображения для которых правильный Content-type обязателен. Поэтому дополним наш код - сначала добавим еще один, небесполезный модуль:

```
let path = require('path'); // подключение модуля для работы с путями в файловой системе
```

Далее создадим объект с mime-типами:

```
let mimeTypeypes = {
    '.js': 'text/javascript',
    '.html': 'text/html',

```

```

    '.css': 'text/css',
    '.jpg': 'image/jpeg',
    '.gif': 'image/gif'
  };

```

Ну а теперь немного изменим код отдачи контента:

```

fs.readFile(pathname, 'utf8', (err, data) => {
  if (err) {
    console.log('Could not find or open file for reading\n');
    response.statusCode = 404;
    response.end();
  } else {
    console.log(`The file ${pathname} is read and sent to the client\n`);
    response.writeHead(200, {
      'Content-Type': mimeTypes[path.extname(pathname)]
    });
    response.end(data);
  }
});

```

Запустите данный код и посмотрите весь ли контент получен браузером?

Задача 3.

Уже лучше, но картинок мы все равно не увидим. Причина проста - это бинарные данные, и читаются они другим способом. Нам немного нужно модифицировать код:

```

http.createServer((request, response) => {
  let pathname, extname, mimeType;
  console.log("Request: " + request.url);

  if (request.url === '/')
    pathname = 'site/index.html';
  else
    pathname = 'site' + request.url;
  extname = path.extname(pathname);
  mimeType = mimeTypes[extname];

  if (extname === ".jpg" || extname === ".gif") {
    try {
      let img = fs.readFileSync(pathname);
      console.log(`The file ${pathname} is read and sent to the client\n`);
      response.writeHead(200, {'Content-Type': mimeType});
      response.end(img);
    } catch (e) {
      console.log('Could not find or open file for reading\n');
      response.statusCode = 404;
      response.end();
    }
  } else {
    fs.readFile(pathname, 'utf8', (err, data) => {
      if (err) {

```

```

        console.log('Could not find or open file for reading\n');
        response.statusCode = 404;
        response.end();
    } else {
        console.log(`The file ${pathname} is read and sent to the client\n`);
        response.writeHead(200, {'Content-Type': mimeType});
        response.end(data);
    }
    });
}
}).listen(8080, ()=>{
    console.log("HTTP server works in 8080 port!\n");
});

```

fs.readFileSync(path[, options]) – синхронное чтение файла, функция возвращает прочитанные данные.

- **path** – строка имя файла (может включать в себя путь к файлу);
- **options** – необязательный параметр, может задаваться в виде строки, тогда он задаёт кодировку файла и прочитанный из файла информация будет представлять собой строку.

1. Запустите код, проверти полную отдачу контента.
2. Добавьте файл favicon.ico в папку site. И научите сервер отдавать и этот тип файлов.